VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

BAKALÁŘSKÁ PRÁCE

Brno, 2019 Radoslav Heriban

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ
DEPARTMENT OF TELECOMMUNICATIONS

GENERÁTOR ZÁTĚŽE A KYBERNETICKÝCH ÚTOKŮ NA PLATFORMĚ FPGA
NETWORK TRAFFIC AND CYBER ATTACKS GENERATOR ON THE FPGA PLATFORM

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Radoslav Heriban**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Tomáš Lieskovan**
SUPERVISOR

BRNO 2019

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní obor **Informační bezpečnost**
Ústav telekomunikací

Student: Radoslav Heriban ***ID:*** 197733
Ročník: 3 ***Akademický rok:*** 2018/19

NÁZEV TÉMATU:

Generátor zátěže a kybernetických útoků na platformě FPGA

POKYNY PRO VYPRACOVÁNÍ:

Téma práce je zaměřeno na generování provozu a kybernetických útoků v TCP/IP sítích na platformě FPGA. Cílem bakalářské práce je seznámení se s platformou FPGA, realizace generátoru zátěže s podporou alespoň tří typů útoků. Studentovo řešení bude vytvořeno pomocí jazyka VHDL nebo P4.

DOPORUČENÁ LITERATURA:

[1] WOLF, Wayne. FPGA based system design. New Jersey: Prentice-Hall, 2004, 530 s. ISBN 0-13-142461-0.

[2] P4 FPGA [Online]. Retrieved September 14, 2018, from http://p4fpga.github.io/

Termín zadání: 1.2.2019 ***Termín odevzdání:*** 27.5.2019

Vedoucí práce: Ing. Tomáš Lieskovan
Konzultant:

prof. Ing. Jiří Mišurec, CSc.
předseda oborové rady

ABSTRAKT

Práca je zameraná na problematiku hardvérovo akcelerovaných DoS útokov. Keďže popularita tohto typu útoku narastá, generátor záťaže kybernetických útokov má za účel slúžiť ako testovací nástroj odolnosti siete. Zvoleným hardvérovým médiom je platforma FPGA, ktorá vďaka svojim vlastnostiam predstavuje ideálny kompromis vlastností pre rapídne prototypovanie a vývoj hardvérových návrhov. V práci je použité vývojové prostredie Xilinx ISE a za jazyk popisujúci požadované správanie FPGA bol zvolený VHDL. Z množstva útokov popísaných v tejto práci boli ďalej implementované a simulované dva z nich - UDP a ICMP záplava. V praktickej časti práce sú spomenuté aj problémy ktorým bolo čelené pre vývojárov, ktorý by chceli podobný projekt realizovať.

KĽÚČOVÉ SLOVÁ

adresa, ASIC, datagram, DoS, FPGA, útok, generátor, ICMP, IP, TCP/IP, ISE, paket, port, protokol, rámec, sieť, Spartan6, UDP, VHDL, Xilinx

ABSTRACT

This thesis is focused on the most common and every day more popular threat of DoS attacks. All networks are vulnerable to this kind of attack, and with growing popularity and intensity it shouldn't be underestimated. The goal of this thesis was creating hardware accelerated generator of DoS traffic intented for testing our own networks and identifying the risks. FPGA technology is used for this task, since it has proven to be more effective way of prototyping hardware design then developing ASIC. The language used to describe desired design behavior is VHDL. Designed ICMP and UDP flood attacks are simulated in Xilinx ISE development environment. Description of problems faced before any result was reached is also included for future researchers interested in similar projects.

KEYWORDS

address, attack, ASIC, datagram, DoS, FPGA, frame, generator, ICMP, IP, TCP/IP, ISE, network, packet, port, protocol, Spartan6, UDP, VHDL, Xilinx

HERIBAN, Radoslav. *Generátor záťaže a kybernetických útokov na platforme FPGA*. Brno, Rok, 42 s. Bakalárska práca. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedúci práce: Ing. Tomáš Lieskovan

VYHLÁSENIE

Vyhlasujem, že som svoju bakalársku prácu na tému „Generátor záťaže a kybernetických útokov na platforme FPGA“ vypracoval samostatne pod vedením vedúceho bakalárskej práce, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedenej bakalárskej práce ďalej vyhlasujem, že v súvislosti s vytvorením tejto bakalárskej práce som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/alebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona Českej republiky č. 121/2000 Sb., o práve autorskom, o právach súvisiacich s právom autorským a o zmene niektorých zákonov (autorský zákon), v znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhej, hlavy VI. diel 4 Trestného zákoníka Českej republiky č. 40/2009 Sb.

Brno ....

podpis autora

POĎAKOVANIE

Rád by som poďakoval vedúcemu bakalárskej práce Pánovy Ing. Tomášovi Lieskovanovi za ochotu pri konzultáciach, spätnú väzbu a hlavne trpezlivosť, ktorú mi preukázal. Ďalej chcem poďakovať svojim rodičom, bez ktorých by som nebol študentom VŠ, hlavne za finančnú a psychickú podporu. Taktiež ďakujem spolužiakom Samuelovi Sidorovi, Martinovi Šuličovi, Antonínovy Bohačíkovi a všetkým známim za morálnu podporu.

Brno . . . .

podpis autora

Obsah

Zoznam obrázkov

Zoznam tabuliek

Úvod

Vývoj harvéru sa odvracá od analógových obvodov k obvodom digitálnym. Jedným z nich sú FPGA čipy, ktoré sa stávajú čoraz populárnejšími, hlavne v odvetví hardvérovej akcelerácie. Vďaka vlastnostiam FPGA kariet ako rapídne prototypovanie, testovanie a možnosti rekonfigurácie, ako aj schopnosťou vysokorýchlostného spracovania dát, sú ideálnou platformou pre generátor záťaže počítačových sietí.

V prvej časti práce sa nachádza teoretické zázemie k pochopeniu princípu, akým sa FPGA čipy programujú, aké majú výhody v porovnaní s inými digitálnymi obvodmi, menovite obvodmi ASIC a mikrokontrolérmi. Obsahuje aj prehľad parametrov zariadení z rodiny Spartan6 od spoločnosti Xilinx, ktorá stojí za vznikom a vynájdením FPGA obvodov. Práca zahŕňa popis VHDL jazyka, ktorý je v súčasnosti najrozšírenejším jazykom na popis správania digitálnych obvodov. Proces návrhu digitálneho obvodu v jazyku VHDL, uvádza kroky ktorými návrh prechádza od zažiatku po implementáciu.

Cieľom tejto práce je navrhnúť generátor záťaže počítačovej siete na platforme FPGA využitím jazyka VHDL. V práci popísaný prenos dát v komunikačných sieťach založený na referenčných modeloch ISO/OSI a TCP/IP. Z vrstiev referenčných modelov sú podrobnejšie popísané protokoly s ktorými je manipulované v praktickej časti k vytvoreniu útokov.

Popularita DoS útokov stále narastá, preto je potrebné vedieť svoju sieť testovať, k čomu bude slúžiť generátor záťaže počítačových sietí. Teoretický úvod obsahuje popis širokého spektra DoS útokov z rôznych vrstiev referenčného modelu TCP/IP, z ktorých niektoré budú implementované v generátore záťaže. Všetky informácie z týchto sekcii slúžia ako teoretický podklad k návrhu generátoru záťaže na platforme FPGA, využívajúc výhody jazyka VHDL.

V druhej časti práce sa nachádza návrh generátoru záťaže. Rovnako popisuje zvolený prístup k riešeniu problematiky a použité nástroje. Ako vývojové prostredie bolo použité ISE od firmy Xilinx. Ako jazyk v ktorom je návrh popísaný bol zvolený VHDL. Generátor má určité obmedzenia, preto boli implementované útoky typu UDP záplava a ICMP záplava.

1 Platforma FPGA

Field Programmable Gate Array (FPGA), v preklade programovateľné hradlové pole, je digitálny obvod podobný mikroprocesorom. Čipy označované týmto pojmom majú univerzálne využitie pretože ich úlohu aj riešenie danej úlohy dokážeme definovať pomocou softwaru, nie výhradne hardwaru, čo znamená že sa dajú aplikovať prakticky kdekoľvek. Samozrejme FPGA nie sú dokonalým riešením každého problému a majú svoje výhody aj nevýhody. V porovnaní s logickými obvodmi navrhnutými konkrétne na riešenie jedinej úlohy (ASIC) sú všestrannejšie, dajú sa vyprodukovať rýchlejšie ako obvody ASIC (to až niekoľko násobne rýchlejšie), no nevýhodami sú napríklad: väčší odber elektriny, cena alebo rýchlosť spracovania dát. Rozdiel medzi ASICmi, FPGA čipmi a mikrokontorlérmi je na prvý pohľad malý. Návrh ASICov je zdĺhavý a navrhnutý hardware sa musí fyzicky vyrobiť. Akonáhle je vyrobený, nedá sa zmeniť (jedine opakovaním celého procesu výroby) a preto nie je vhodný na prototypovanie. Čas od návrhu ASICu po jeho produkciu je zhruba 6 mesiacov. FPGA čipy obsahujú množstvo súčiastok ktorých prepojenie dokážeme definovať softwarovo, čo umožňuje rýchle prototypovanie. Táto vlastnosť ďalej umožňuje rapídne zmeny v implementácii bez akýchkoľvek nákladov. Výhody FPGA čipov sú jasné z nasledujúcej tabuľky 1.1.

Tab. 1.1: Porovnanie vlastností FPGA a ASICov.

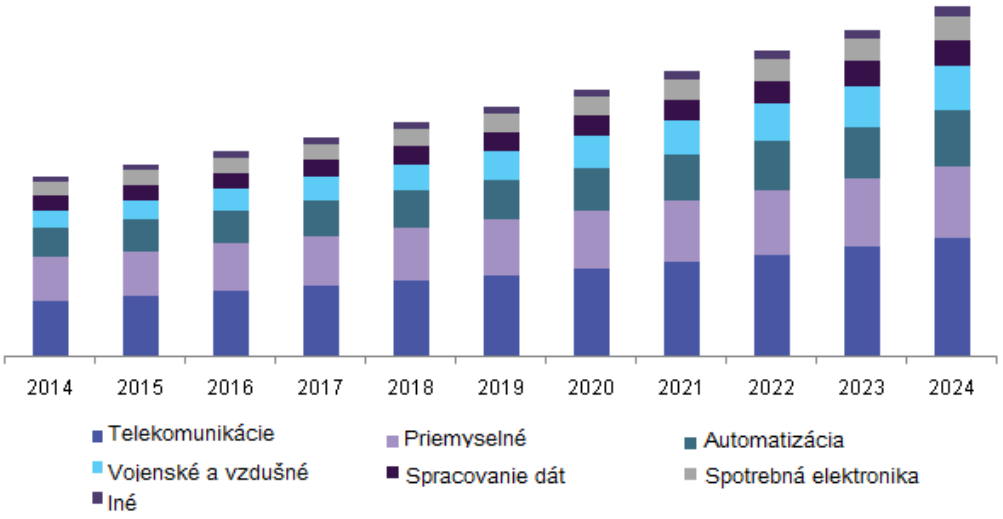
	FPGA	**ASIC**
Plynulosť návrhu	Jednoduchší	Veľmi komplexný
Cena	Stredná	Vysoká
Výkon	Stredný	Vysoký
Spotreba energie	Vysoká	Nízka
Počet logických hradiel	Stredná	Nízka
Doba implementácie	Krátka	Dlhá

Rozdiel medzi mikrokontrolérmi a FPGA čipmi je hlavne v tom, že mikrokontroléry obsahujú pamäť s inštrukciami ktoré sú posielané do CPU. Toto spracovanie je sekvenčné, kdežto po naprogramovaní obsahujú FPGA čipy logický obvod určený na riešenie konkrétnej úlohy a inštrukcie sa spracúvajú paralelne. Obrovskou výhodou FPGA čipov je možnosť opakovaného využitia.

Kedysi boli FPGA považované len za akési „logické medziprvky“ alebo zariadenia určené výhradne na prototypovanie. V dnešnej dobe sú bežnými súčasťami komplexnejších prvkov, napríklad vysokorýchlostných telekomunikačných zariadení a stali sa bežnou platformou pre implementáciu digitálnych zariadení. V budúcnosti sa

predpokladá ešte častejšie využívanie FPGA kôli Mooreho zákonu o dvojnásobnom zmenšovaní tranzistorov každých 18 mesiacov. Zatiaľ čo cena FPGA čipov môže rásť lineárne s počtom tranzistorov, cena produkcie ASICov rastie exponenciálne pretože každý dizajn je unikátny, pričom FPGA čipy sú normalizované [1].

FPGA sa osvedčili v hardvérovej akcelerácii algoritmov, napríklad v šifrovaní, Microsoft ich uplatňuje v akcelerácii učenia umelej inteligencie aj v Bing vyhľadávači [2], našli uplatnenie v odvetviach ako vzdušná obrana, automatizácia, medicína, audio inžinierstvo, spracovanie videa a obrázkov, bezdrôtové komunikácie, akcelerácia v dátových centrách či v spotrebnej eletronike. Príkladom je graf využitia FPGA v Spojených štátoch Amerických spolu s odhadom na budúce roky na obrázku 1.1[3].

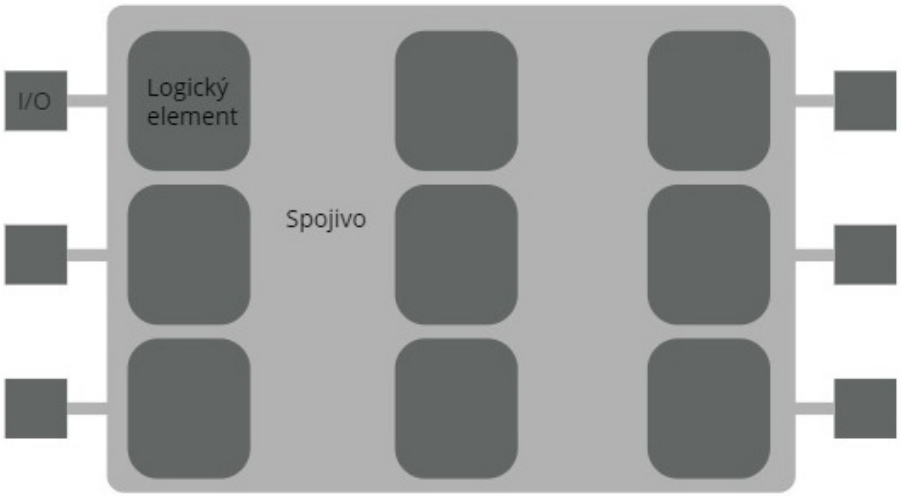


Obr. 1.1: Využitia FPGA v Spojených štátoch amerických.

1.1 Architektúra FPGA

Základom všetkých FPGA čipov sú minimálne tieto tri komponenty: kombinačná logika, spoje a I/O vývody. Kombinačná logika je implementovaná každým výrobcom inak no najčastejšie pomocou menších jednotiek nazývaných logické elementy (Altera) alebo vyhľadávacie tabuľky (Xilinx). Skupina niekoľkých logický elementov v kombinácii s registrom tvorí logický blok. Spoje medzi logickými blokmi sú takisto programovateľné. Klasicky sú všetky bloky prepojené maticovo, pričom jeden čip môže obsahovať viacero vrstiev respektíve sietí prepojení. Schému zapojenia zázorňuje obrázok 1.2. Príkladom môže byť oddelená sieť spojiva pre hodinové signály a

podobne. Programátor má takisto kontrolu nad I/O výstupmi pre ktoré môže určiť ich správanie – buď vstup, výstup, alebo oboje.



Obr. 1.2: Základná architektúra FPGA.

Všetky z týchto komponentov potrebujú konfiguráciu. Existuje viacero techník konfigurácie. Najčastejšie používané technológie sú SRAM, antifuse a FLASH [1]. Parametre, na základe ktorých vieme určiť či je čip vhodný pre našu aplikáciu zahŕňajú hlavne:

- počet I/O vývodov,
- množstvo logických blokov,
- spotreba elektrickej energie,
- cena,
- typ uloženia konfigurácie.

1.1.1 Typy FPGA

Od minulosti prešli FPGA čipy značným vývojom. Staršie typy ukladania konfiguračných súborov zahŕňajú napr. EPROM založené na technológii CMOS, ktoré sa už v dnešnej dobe nepoužívajú. Ich nevýhodou bolo jednorázové programovanie. Rovnako tak zastaralým typom je aj uloženie na báze Fuse a PROM.

FPGA na báze SRAM

Tento typ FPGA má prídavnú statickú pamäť s ktorou je prepojený. V nej sa uchováva konfigurácia, ale iba po dobu kým je čip napájaný pretože pamäte typu SRAM sú nestále. Riešením je pridanie ďalšieho prvku, ktorý si konfiguráciu uchová aj bez napájania a slúži ako záložné úložisko konfigurácie z ktorého sa po obnovení napájania konfigurácia nahrá do SRAM. V tomto prípade vzniká oneskorenie načítania konfigurácie do FPGA čipu v rádoch stoviek milisekúnd. Výbornými vlastnosťami SRAM FPGA sú: možnosť rekonfigurácie a to aj počas behu a nízka spotreba energie [5].

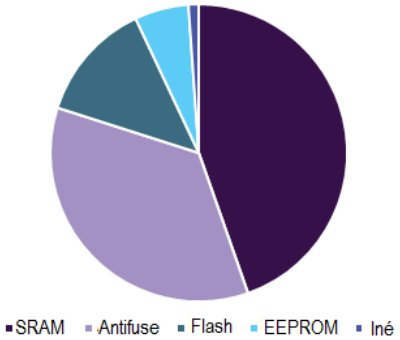
FPGA na báze FLASH

Pamäť typu FLASH je založená na technológii EEPROM . Uchováva si zapísanú konfiguráciu aj pri výpadku napájania. Výhodou tejto technológie je väčšia odolnosť proti radiácii a ochrana bitového toku konfigurácie pred neautorizovaným čítaním. Rovnako ako SRAM podporuje rekonfiguráciu FPGA aj za behu [1].

FPGA na báze Antifuse

Technológia antifuse pracuje na opačnom princípe ako poistka. Antipoistka je elektronická súčiastka s vysokým elektrickým odporom, ale po prekročení určitej hranice - programovacieho napätia, sa jej odpor zmenší a stane sa priepustnou. Každá antipoistka je teda programovateľná len jeden krát. Neprekvapivo aj FPGA čipy využívajúce túto technológiu môžeme naprogramovať len raz. Výhodou je nulové omeškanie načítania konfigurácie, pretože k žiadnemu načítavaniu nedochádza. Konfigurácia je definovaná priechodnými cestami v spojive logických blokov [6].

Najčastejšie používané technológie ukladania konfiguračných súborov na FPGA čipoch znázorňuje obrázok 1.3[3].



Obr. 1.3: Pomer najviac využívaných typov pamäte v roku 2015.

1.2 Xilinx Solutions

Spoločnosť Xilinx vyvíja vysoko flexibilné and prispôsobivé platformy, ktoré umožňujú rapídnu inováciu napriek množstvom technológii. Xilinx je objavitelom FPGA, hardvérovo programovatelných systémov na doske a adaptatívnych výpočet akcelerujúcich platform navrhnutých k doručeniu najdynamickejšej procesorovej technológie v svojom obore.

Xilinx má na trhu množstvo FPGA čipov rozdelených do rodín. Najaktuálnejšími sú Spartan6, Spartan7, Virtex7, Kintex7, Artix7, Kintex UltraScale, Kintex UltraScale+, Virtex UltraScale a Virtex UltraScale+.

Zariadenia z rodiny Spartan6 využívajú polovodiče s veľkosťou 45nm. Uživatelom ponúkajú vysoký počet vstupných a výstupných hradiel a sú ideálnym riešením pre zložitejšie návrhy. Táto rodina má tiež vlasné vývojové prostredie ISE Design Suite. Tabuľka 1.2 obsahuje zoznam čipov z rodiny Spartan6 a ich vybrané parametre.[7]

Tab. 1.2: Prehľad vybraných parametrov zariadení z rodiny Spartan6.

Zariadenie	Logické buňky	Konfigurovatelné logické bloky	závory	Max. uživatelské I/O
XC6SLX4	3,840	600	4,800	132
XC6SLX9	9,152	1,430	11,440	200
XC6SLX16	14,579	2,278	18,224	232
XC6SLX25	24,051	3,758	30,064	266
XC6SLX45	43,661	6,822	54,576	358
XC6SLX75	74,637	11,662	93,296	408
XC6SLX100	101,261	15,822	126,576	480
XC6SLX150	147,443	23,038	184,304	576
XC6SLX25T	24,051	3,758	30,064	250
XC6SLX45T	43,661	6,822	54,576	296
XC6SLX75T	74,637	11,662	93,296	348
XC6SLX100T	101,261	15,822	126,576	498
XC6SLX150T	147,443	23,038	184,304	540

1.3 Programovací jazyk VHDL

Patrí medzi programovacie jazyky vyššej úrovne. Vyššie programovacie jazyky poskytujú určitú abstrakciu, ktorá robí jazyk podobnejší ľudskej reči. VHDL je jazyk určený na popis a simuláciu číslicových obvodov, teda hardvéru.

Existuje viacero jazykov s rovnakým účelom. Spoločne sa označujú ako Hardware Description Language – HDL (jazyk popisu hardvéru). Jedným z nich je aj Verilog, ktorý je veľmi podobný VHDL a má podobné vyjadrovacie schopnosti. V praxi sa môžeme stretnúť s hybridným systémom, ktorého časť je popísaná jazykom VHDL a druhá časť vo Verilogu. V Ázii a Spojených Štátoch je viac využívaný Verilog, zatial čo v Európe VHDL. Pri práci s HDL je nutné aby si programátor uvedomoval, že píše program ktorý bude najprv preložený do strojového kódu iba na testovacie účely, ale neskôr bude implementovaný priamo pomocou logických hradiel. Proces generovania číslicového obvodu na základe popisu vo VHDL sa nazýva syntéza. Základná knižnica jazyka VHDL obsahuje klasické dátové typy ako integer a podobné, ale umožňuje aj definíciu vlastných dátových typov. Niektoré konštrukcie jazyka VHDL sú určené len pre simulované prostredie, a nie je možné ich syntetizovať do logických hradiel. Jednou z týchto výnimiek je napríklad konštrukcia file. Existuje viacero syntetizátorov a niektoré syntetizátory majú lepšie prekladové vlastnosti ako iné, ale vo všeobecnosti sú si veľmi podobné. Jazyk VHDL dokáže zvládnuť paralelnú povahu programu implementovaného hardvérovo ale poskytuje aj možnosť využiť konštrukciu process. Kód v procese je vykonávaný sekvenčne. Má pevne definované dátové typy a nerozlišuje medzi veľkými a malými písmenami. Proces tvorby programu vo VHDL sa skladá z hlavného programu a testovacieho programu. Simulátor jazyka používa signály definované v testovacom kóde ako vstup do hlavného kódu pričom kód testovacieho súboru nemusí byť syntetizovateľný pretože sa nesyntetizuje [8]. Hlavnými vlastnosťami HDL jazykov sú:

- nezávislosť na hardwari kde bude kód použitý, nezáleží na výrobcovi čipu, prípadne môžeme na základe tohto kódu vyrobiť ASIC,
- možnosť simulácie programu a písania vlastných testovacích programov,
- voľne prístupný zdrojový kód, vďaka ktorému existuje viacero vývojových prostredí a syntetizátorov,
- kompatibilita s ostanými HDL jazykmi, čo umožňuje využitie najlepších vlastností viacerých jazykov.

Najzákladnejšími návrhovými jednotkami sú: entita, architektúra a proces. Entita je konštrukcia, pomocou ktorej definujeme blok - súčiastku - s vstupnými a výstupnými signálmi. Vstupy a výstupy sú definované v sekcii port. Smer signálu môže nadobúdať hodnoty in, out, inout, buffer a linkage. Správanie, teda spracovanie jednotlivých

signálov vnútri entity popisuje jej architektúra. Každá entita musí mať minimálne jednu architektúru, ale entity môžu mať niekoľko architektúr a v module kde entitu použijeme je možné určiť, ktorá architektúra definuje jej správanie. Každý zdrojový súbor by mal obsahovať jednu entitu a jej architektúru/y. Jazyk VHDL taktiež obsahuje knižnice, ktoré je možné do modulov importovať použitím kľúčového slova library nasledovaného názvom knižnice (napríklad std_logic_1164 - knižnica na implementáciu logických dátových typov). Moduly je možé spájať alebo používať v nadradenom module. Tento proces je možný vďaka instanciácii modulu. Ten sa stane komponentom v nadradenom module. [8]
Proces, ktorým každý hardvérový návrh musí prejsť pozostáva z nasledujúcich krokov:

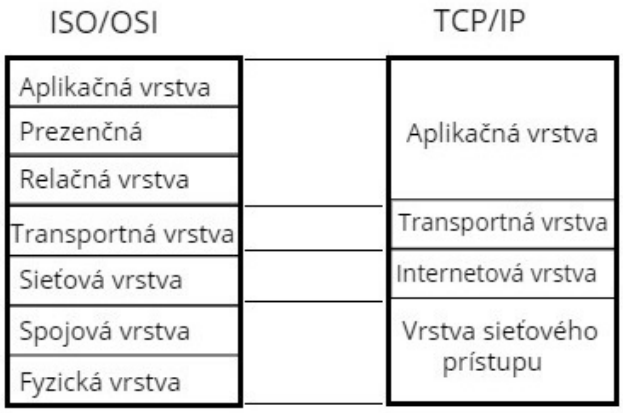
1. kontrola sytaxu kódu - slúži na odhalenie chýb v zdrojovom kóde,
2. syntéza - prevod z popisu očakávaného správania do logckých hradiel, ktoré toto správanie zaručia vo fyzickej implementácii,
3. generovanie programovatelného súboru - takzvaný bitstream, ktorý je v ďalšom kroku nahraný do pamäte FPGA čipu,
4. konfigurácia zariadenia - proces samotného nahrania bitstreamu do pamäte čipu,
5. simulácia - volitelný krok na testovanie správania navrhnutého obvodu, ktorý môžeme spustiť buď pred alebo po syntéze.

1.4 Prenos dát v telekomunikačných sieťach

Moderné komunikačné siete poskytujú svojim účasníkom radu služieb, napríklad: schopnosť prenášania dát, či už textových, zvukových, obázkov alebo videa, smerovanie dát medzi účasníkmi, fyzické spojenie účasníkov, údržbu siete a podobne. Spôsob, akým sú dáta v sieti prenášané môže mať dvojaký charakter - spojový a nespojový. Spojový prenos funguje na princípe potvrdenia o nadviazaní spojenia medzi dvojmi účasníkmi kdežto nespojový prenos prebieha jednostranne, kedy odosielateľ odošle dáta bez potvrdenia spojenia príjmateľom. Oba typy prenosu môžu byť ďalej klasifikované na spoľahlivý alebo nespoľahlivý prenos. V prípade, že využívame spoľahlivý prenos dát, príjemca informuje odosieľatela o tom, či obdržal všetky dáta. Ak príjmatel nejaké dáta neobdrží, odosielateľ ich zašle znova. Proces sa opakuje kým niesú doručené všetky dáta. V kontraste, nespoľahlivá komunikácia nerieši obdržanie dát na strane príjemcu.

Základnými modelmi sieťovej architektúry sú všeobecnejší model ISO/OSI (Open System Interconection), ktorý popisuje komunikáciu otvorených systémov a model

TCP/IP. Oba modely znázornené na obrázku 1.4 popisujú a štandardizujú prenos dát v telekomunikačných sieťach s rozmanitými zariadeniami. Model TCP/IP je navrhnutý priamo pre komunikáciu počítačov a iných zariadení v sieti Internet. Tá je možná na základe protokolov, a takzvaných vrsiev do ktorých sú rozložené. Modely sú štrukturované do vrstiev ktoré pokrývajú všetky aspekty dátového prenosu od služieb pre aplikácie po fyzický prenos medzi dvoma koncovými bodmi. Protokolové vrstvy sa dajú implementovať softvérovo aj hardvérovo. Kedže protokoly fyzickej a spojovej vrstvy riešia prenos dát fyzickým vodičom, najčastejšie sú implementované v sieťových kartách, ktoré sú s nosným médiom - najčastejšie medeným káblom alebo optickým vláknom spojené a umožňujú ďalšiu manipuláciu s dátami. Všetky protokoly použité na prácu s dátami v danom zariadení nazývame protokolový zásobník[13].



Obr. 1.4: Referenčné modely ISO/OSI a TCP/IP.

1.4.1 Fyzická vrstva

Je najnižšou vrstvou v oboch referenčných modeloch Jednotkami, s ktorými fyzická vrstva pracuje sú bity, ktoré sú v prípade káblu reprezentované napätím na vodičoch. Úlohou fyzickej vrstvy je prenos jednotlivých bitov dát medzi dvoma bodmi. Spôsob prenosu bitov určuje použitý protokol. Pre krútený-párový medený kábel existuje niekoľko možných riešení v protkolovej sade Ethernet. Taktiež špecifikuje rozhrania a konektory ktorými sú fyzické nosiče pripojené k zariadeniu[13].

1.4.2 Spojová vrstva

Dáta putujú sieťou zvyčajne mnohými uzlami než prídu do svojej destinácie, preto táto vrstva slúži na prenos dát medzi jednotlivými uzlami v jednej sieti. Jednotka s

ktorou spojová vrstva operuje sa nazýva rámec. Štandard IEEE 802 ďalej rozdeluje spojovú vrstvu na MAC (Medium access control) vrstvu, ktorá riadi prístup k médiu a LLC (Logical Link) vrstu zabezpečujúcu korekciu chýb. Spolu s fyzickou vrstvou tvorí vrstvu sieťového pripojenia referenčného modelu TCP/IP [13].

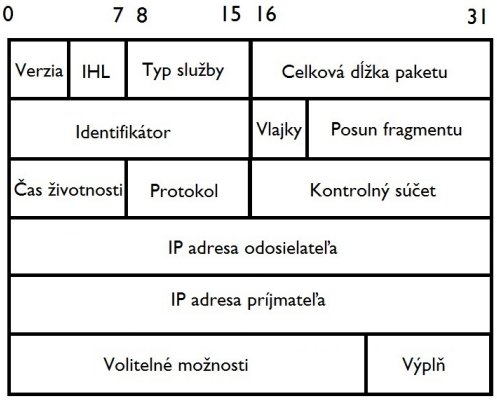
1.4.3 Sieťová vrstva

Taktiež známa ako Internetová vrstva v modeli TCP/IP, na rozdiel od spojovej vrstvy prenáša dáta medzi uzlami v rozdielnych sieťach. Jednotkou prenosu je datagram. Najznámejším protokolom sieťovej vrstvy je IP Protokol, ktorý dátam pridáva rôzne informácie potrebné na prenos medzi sieťami, vrátane adries [13].

Protokol IP

Je protokol sieťovej vrstvy určený pre prenos datagramov medzi dvoma projenými sietami. Datagramy putujú zariadeniami ako smerovače pripojenými do siete a sú preposielané ďalej na základe interpretácie ich cieľovej internetovej adresy. Adresovanie je hlavným mechanizmom IP protokolu.

Adresa je 32 bitový identifikátor fixnej dĺžky. Začína číslom siete nasledovaným lokáloou adresou. Adresy sa formálne rozdelujú na triedy znázornené na obrázku. Z dvôvodu konečného počtu IP adries, ktorý nie je dostatočne veľký na to aby poskytol každému zariadeniu unikátnu IP adresu existujú dve verzie protokolu IP. Verzia IPv4 využíva menší adresný priestor v kombinácii s technikou sieťových masiek, zatial čo verzia IPv6 poskytuje rozšírenie adresného priestoru v rozsahu ktorý umožnuje každému zariadeniu priradiť unikátnu adresu v celkom priestore siete Internet. Na obrázku 1.5 môžeme vidieť dáta obsiahnuté v IP hlavičke.



Obr. 1.5: Hlavička protokolu IP

Sieťová vrstva pridáva dátam hlavičku IP protokolu. Hlavička pozostáva z 24 oktetov dát ktoré môžeme vidieť na obrázku. Polia v hlavičke nesú následujúce informácie:

- Verzia – indikuje formát hlavičky. IPv4 má hodnotu 4, IPv6 hodnotu 6.
- IHL – dĺžka internetovej hlavičky. Táto hodnota slúži ako ukazovatel na začiatok dát.
- Typ Služby – predstavuje abstraktný parameter požiadavku akú kvalitu služby vyžadujeme. Siete poskytujú službám zvýhodnené smerovanie, založené na kompromise medzi nízkym omeškaním, vysokou spoľahlivosťou a vysokým prietokom.
- Celková dĺžka – je dĺža datagramu vrátane internetovej hlavičky a dát meraná v oktetoch. Maximálna dĺžka je 65535 oktetov. Klasická dĺžka datagramov je 576 oktetov.
- Identifikátor – potrebný pri skladaní fragemntovaných dát.
- Čas k životu – čas, určujúci ako dlho môže paket zotrvať v obehu siete. Každé zariadenie, ktoré paket spracováva musí túto hodnotu znížiť o jedna. V prípade, že hodnota je rovná nule, paket musí byť zničený. Slúži na ničenie blúdnych paketov.
- Protokol – určuje zapúzdrený protokol umiestnený v dátovej časti. Napríklad číslo protokolu ICMP je 1. Čísla protokolov sú fixne ustanovené.
- Kontrolný súčet hlavičky - kontrola integrity hlavičky.
- Zdrojová adesa – adresa odosielateľa.
- Cieľová adresa – adresa príjmateľa.
- Možnosti – volitelné hodnoty, napr. "Security".
- Výplň – v prípade malého počtu dát vyplní hlavičku do dĺžy delitelnej 32.

V procese smerovania správy z jedného zariadenia do druhého môže nastať prípad, kedy je veľkosť datagramu vačšia ako maximálna povolená veľkosť paketu v sieti. Tento problém rieši další mechanizmus - fragementácia[11].

Protokol ICMP

Internet Control Message Protocol je súčasťou IP protokolu. ICMP správy slúžia na oznámenie poruchy alebo analýzy stavu zariadnie, napr. hosťa alebo brány. Protokol ICMP používa podporu IP protokolu a na prvý pohľad vypadá ako nadradený protokol, no v skutočnosti je súčasťou IP a musí byť implementovaný v každom zariadení. Počas prenosu dát môže nastať niekolko nepredvídateľných situácii, napr. hosť je nezastihnutelný alebo brána je zahltená. ICMP správy oznamujú chbyby v spracovaní datagramov pridaním ICMP hlavičky uvedenej na obrázku 1.6[12].

Hlavička obsahuje následovné polia:

Obr. 1.6: Hlavičky protokolu ICMP Echo Request/Reply.

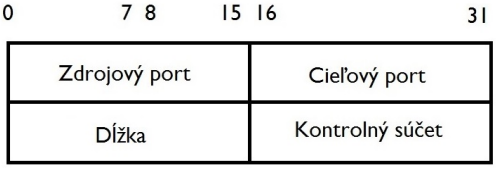
- Typ – označuje typ ICMP správy. Napr. pre Echo Request je hodnota 8, pre Echo Reply hodnota 0.
- Kód – špecifickejsie definuje typ správy.
- Kontrolný súčet – hodnota na overenie integrity dát.
- Identifikátor – s kombinácii so sekvenčným číslom slúži na párovanie požiadavkov a odpovedí.
- Sekvenčné číslo – dodatokový indikátor k identifikátoru.

1.4.4 Transportná vrstva

Zabezpečuje prenos aplikačných dát medzi dvoma zariadeniami. Dokáže dáta segmentovať/desegmentovať, taktiež spracovaváť dáta premenlivej veľkosti, kontrolovať integritu dát či doručovať segmenty v poradí. Transportná vrstva používa jeden z dvoch možných protokolov, UDP pre nespojovú komunikáciu, TCP pre spojovú. Protokol TCP kontroluje a riadi rýchlosť prenosu dát príjmateľa a odosielateľa. Rozkladá veľké množstvo dát na menšie jednotky - segmenty. UDP protokol je nespojový, to znamená, že dáta jednoducho posiela a nekontroluje doručenie ani ich poradie[14].

Protokol UDP

Je protokol transportnej vrstvy. Jeho pomocou sú dáta odoslané príslušnej aplikácii. Je vnorený do dátovej časti IP paketu. Hlavička UDP protokolu ilustrovaná obrázkom 1.7 ukazuje jeho jednoduchosť[10].



Obr. 1.7: Hlavička protokolu UDP.

Hlavička pozostáva z informácii:

- Zdrojový port – volitelný údaj o aplikácii ktorá dáta odoslala. Ak je neuvedený, hodnota je 0. V prípade, že je hodnota nenulová, predpokladá sa, že na tento port má bzť smerovaná odpoveď.
- Cieľový port – identifikuje aplikáciu pre ktorú sú dáta určené.
- Dĺžka – počet oktetov datagramu vrátane UDP hlavičky a dát.
- Kontrolný súčet – slúži na odhalenie chybne prijatého paketu.

1.4.5 Aplikačná vrstva

Je najvyššou vrstvou oboch referenčných modelov. Aplikácie sú rozložené na viacerých zariadeniach a dáta si medzi sebou predávajú správami. Protokoly tejto vrstvy sú napríklad HTTP, FTP, SMTP[14].

1.5 DoS útoky

V kyberpriestore môžeme hovoriť o rôznych útokoch, ktoré sa dajú klasifikovať podľa ich mechanizmu a následkov. Najznámejším typom útoku je zrejme phishing, ktorý často slúži iba ako šíriteľ iného útoku. Pod pojmom „útok“ nás najskôr napadá krádež dát alebo neoprávnené používanie systému. Môže však nastať situácia, kedy sú dáta v bezpečí, ale nevieme sa k ním dostať, alebo špičkovo zabezpečený webserver neodpovedá na žiadosti klientov. Tento typ útoku sa nazýva Denial of Service – bránenie v používaní služby. Službou môže byť napríklad používanie aplikácie alebo schopnosť funkcionality sieťového prvku. Cieľom útočníka je zabrániť legitímnym užívateľom v používaní služby za využitia všetkých jemu dostupných prostriedkov z užívateľskej strany interakcie.

V dnešnej dobe je DoS spájaný hlavne so záťažou hardwarových prvkov počítačovej siete. V súčasnosti je málo pravdepodobné, že jediný užívateľ dokáže zahltiť server, preto sa skôr stretneme s výrazom DDoS – distribuované bránenie vo využívaní služby. V tomto prípade útočník využíva viacero strojov v koordinovanom útoku. Príkladom DoS útoku dnešnej doby je znemožnenie pripojenia alebo udržania spojenia s WiFi prístupovým bodom prostredníctvom odosielania deautentikačných paketov. Vo všeobecnosti vieme DoS útoky rozdeliť na dva typy: tie ktoré službu (server, program, sieťový prvok) zastavia úplne a také, ktoré službu extrémne spomalia zahltením množstvom dotazov. Najnovším trendom je DDoS ako služba, preto narastá počet DDoS útokov.

Najväčší dokumentovaný DDoS útok v súčasnosti zahlcoval v roku 2018 servery

platformy GitHub dátovým tokom až 1,3 terabitov za sekundu [15]. DDoS útoky sa stávajú čoraz populárnejšími. Podľa štatistýk počet aj sila incidentov každý rok rapídne narastá. Trendom sú krátke zato intenzívne útoky [16]. Popis všetkých následovných typov útokov pochádza z rovnakého zdroja, preto je uvedený len na konci.

1.5.1 Útoky na báze dátového objemu

Medzi tento typ útokov spadajú tzv. záplavy (floods) napr. UDP záplava, ICMP záplava a iné záplavy falošnými paketmi. Cieľom útočníka je preťaženie linky obete. Sila útoku je meraná v bitoch za sekundu (Bps).

UDP Flood

UDP záplava je definovaníciou akéhokoľvek DDoS útoku, ktorý posiela obeti neobvykle veľké množstvo UDP paketov. Cieľom toho útoku je zahltenie náhodných portov obete. Následne sa systém obete snaží zistiť aká aplikácia naslúcha danému portu a v prípade, že nenájde žiadnu, odpovedá útočníkovy ICMP paketom "Cieľ je nedostupný". Tento proces vyťažuje zdroje obete, čo môže viesť až k celkovej nedostupnosti cieľa útoku.

DNS Flood

Je typ UDP záplavy mierený konkrétne na DNS servery. Tie sú zasypané množstvom požiadavkov z mnohých IP adries. Server nedokáže rozoznať útočníkove pakety, pretože niesú nijak odlišné od normálnej premávky. Pakety sú usporiadané presne tak, ako v prípade legitímneho DNS požiadavku. Útok vyčerpá sieťové zdroje servera a zahltí jeho pásmo. Tento typ útoku je veľmi ťažko detekovateľný a dokáže prekonať detektory monitorujúce anomálie v premávke.

ICMP (Ping) Flood

Rovnako ako v prípade UDP záplavy, aj ICMP záplava vyťažuje zdroje cieľa prostredníctvom ICMP Echo žiadostí (ping), zvyčajne čo najrýchlejším zasielaním čo najväčšieho množstva paketov bez čakania na odpovede. Tento typ útoku zahlcuje rovnako upload aj download šírky pásma, pretože obeť pakety príjma a zároveň na ne odpovedá, čo spôsobuje značné spomalenie systému.

IP Null Attack

Útočník odosiela pakety obsahujúce upravené pole IP hlavičky, ktoré špecifikuje použitý transportný protokol. V tomto prípade je použitá hodnota NULL, čo má často za následok že firewally takýto paket prepustia ako neklasifikovaný. Ajked v

dnešnej dobe je null hodnota rezervovaná pre IPv6 "Hop-by-hop Option", nie každý server dokáže bezpečne takýto paket spracovať. Veľké množstvo paketov tohto typu zahltí systémové zdroje na analýzu a znefunkční ho.

1.5.2 Protokolové útoky

Zahrňujú SYN záplavu, útoky roztrieštenými paketmi, ping smrti, smurf DDoS a podobne. Tento typ útoku zahlcuje priamo server alebo iné prvky v sieti ako napríklad firewally, záťažové vyvažovače, proxy servery a pod. Sila útoku je meraná v paketoch za sekundu (Pps).

TCP Null Attack

Tento typ útoku je využívaný na skenovanie portov. Útok je špecifický paketmi obsahujúcimi prázdne hodnoty TCP vlajok, teda neobsahuju žiadnu z informácii URG, ACK, PSH, RST, SYN, FIN.

SYN Flood

SYN záplava zneužíva vlatnosti TCP spojovacej sekvencie three-way-handshake (trojcestné potrasenie rukou), v ktorej žiadateľ odošle SYN žiadosť o nadviazanie TCP spojenia, na ktorú musí obdržať SYN-ACK odpoveď od svojho partnera a v poslednom kroku odošle žiadateľ ACK odpoveď. V prípade útoku, žiadateľ opakovane zasiela iba SYN žiadosti a neodpovedá na SYN-ACK žiadosti svojej obete. Druhou variantou je zasielanie SYN žiadostí s falošnou zdrojovou IP adresou. V obidvoch prípadoch sa obeť márne snaží nadviazať TCP spojenie na viacerých portoch, až dokým sa nevyčerpajú všetky voľné porty a žiadne dalšie pripojenie (ani legitímne) nie je možné.

SYN-ACK Flood

Za bežného fungovania server odpovedá na SYN požiadavky SYN-ACK odpoveďou. Útok je veľmi podobný SYN záplave. V tomto prípade je na server obeťe odosielané množstvo SYN-ACK paketov a server sa snaží na základe tabuľky spojení vyhodnotiť každý prijatý paket. Obmedzené zdroje servera ako RAM alebo CPU nedokáže vydržať nátlak tohoto útoku.

RST/FIN Flood

Ďalší z rodiny útokov zneužívajúcich vlastností TCP spojenia. Obeť, zvyčajne server, je zahltený RST/FIN paketami ktoré nekorešpondujú so žiadnou reláciou v tabuľke aktívnyhc spojení.

Ping of Death

Ping smrti spočíva v tom, že útočník posieľa zdeformované ICMP pingy obeti. Maximálna dĺžka IPv4 paketu vrátane hlavičky je 65535 bytov, avšak spojová vrstva často určuje limity maximálnej veľkosti rámca, napríklad v technológii Ethernet je to 1500 bytov. V takom prípade sa veľký IP paket rozdelí na niekoľko menších paketov známych ako fragmenty a recipient si fragmenty spätne zloží do jedného uceleného paketu. Tento proces sa dá zneužiť zaslaním špeciálne upravených fragmentov, ktoré po spätnom zložení presahujú veľkosť 65535 bytov. To môže viest k pretečeniu pamäte alokovanej pre daný paket, čo poškodí iný, ideálne legitímny paket.

NTP Amplification

Tento útok využíva verejne známe a prístupné servery na synchronizáciu času. Network Time Protocol (NTP) servery preťažia obeť UDP premávkou. Názov "NTP zosílenie" dostal kôli pomeru požiadavok ku odpovedi, ktorý je v tomto prípade v rozmedzí od 1:20 do 1:200, niekedy aj viac. To znamená, že útočník so zoznamom NTP serverov, ktorý môže zostaviť pomocou nástrojov ako Metasploit alebo Shodan, dokáže ľahko vygenerovať zničujúcu premávku na linke obeťe a zahltiť celé pásmo.

1.5.3 Útoky na aplikačnej vrstve

Do tejto kategórie je možné zaradiť "low-and-slow" útoky, HTTP GET/POST záplavy, útoky špecificky zamerané na slabiny v serverových technológiach napr.: Apache, alebo operačných systémov napr.: Windows, OpenBSD a iné. Požiadavky zasielané útočníkom na prvý pohľad pôsobia legitímnyn dojmom, ale ich cieľom je vynútenie vypnutia serveru. Veľkosť útoku je meraná v HTTP požiadavkoch za sekundu (Rps).

HTTP Flood

Na rozdiel od ostatných útokov, HTTP záplava nevyužíva zdeformované pakety, ani falošné požiadavky alebo odrazové techniky ale relatívne legitímne HTTP GET alebo POST požiadavky a útočník si vystačí s menšou sírkou pásma ako pri iných útokoch Útok je najefektívnejší keď prinúti obeť odoslať maximálne možné množstvo dát v odpovedi na každú žiadosť. Tento útok zahlcuje hlavne linku obete.

Recursive HTTP GET Flood

Útočník si na začiatok vyžiada niekoľko stránok a analyzuje ich obsah. Potom rekurzívne od servera požaduje všetky nájdené objekty. Tento typ útoku je kombinovateľný s inými HTTP založenými útokmi, pretože v spojení vytvára dojem legitímneho prehliadania stránky a je tažko detekovatelný. Zamedziť sa mu dá zmenšením sírky pásma alokovanej každému pripojeniu, čo má za efekt spomalenie načítavania stránky.

Random Recursive GET Flood

Častou obeťou tohto útoku sú fóra s číslovanými stranami. Automatizovaný bot posiela na server požiadavky s náhodným číslom stránky čím vytvára dojem legitímneho užívania služby.

SIP Malformed Attack

Je zameraný na znemožnenie fungovania SIP serverov. Útočník odosiela neštandardné správy so schválne neplatnými dátami. Toto môže mať za následok znestabilnenie serveru, ktoré v spojitosti s iným útokom môžu server úplne znefunkčniť.

SIP Register Flood

Spočíva v odosielaní veľkého množstva SIP REGISTER alebo SIP INVITE paketov na SIP server. Server zlyhá následkom vyztaženia systému alebo zahltením šírky pásma.

Slowloris

Slowloris je špecifický útok, ktorý dokáže zahltiť iba jedinú službu – webserver, bez akéhokoľvek zásahu do ostatnej funkcionality napadnutého systému. Toto správanie je možné vďaka otváraniu spojení iba na portoch webserveru (najbežnejsie na portoch 80, 8080, 443) a ich kontinuálnom udržiavaní pri živote po čo najdlhšiu dobu. Slowloris nadviaže spojenie s webserverom a potom zasiela neúplné HTTP žiadosti. Žiadosti obsahujú HTTP hlavičky, ale neodsahujú legitýmnu žiadosť o dáta zo serveru. To spôsobí, že na strane serveru je otvorených veľa spojení, ktoré sa obnovujú ale nezanikajú alebo nezanikajú dostatočne rýchlo, a kedže server môže nadviazať len obmedzený počet spojení, časom sa nevyhnutne dostane do stavu, kedy nadviazal maximálny počet spojení, čo znemožní legitýmnym užívateľom prístup [17][18].

2 Programové riešenie

Cieľom práce je implementovať tri DoS útoky za vuyžitia jazyka VHDL. Pre návrh bolo z dvojice Intel Quartus a Xilinx ISE zvolené vývojové prostredie Xilinx ISE. Prostredie priamo podporuje vývoj pre čipy z populárnej rodiny Spartan6. Generátor nie je navrhnutý na konkrétny čip pretože zaberá primerané množstvo zdrojov FPGA takže je možné použiť aj staršie čipy.

Samotný generátor je závislý na funkčnej sieťovej komunikácii FPGA s inými zariadeniami teda cieľami útokov. Jedným z cieľov návrhu bolo prenositelnosť a nezávislosť na iných komponentoch, ktoré sú často prídavnými a volitelnými nie základnými súčasťami všetkých FPGA systémov, inými slovami všetko potrebné malo byť obsiahnuté v samotnom čipe.
Požiadavky na funkčný generátor spočívali z:

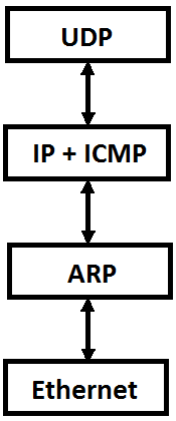
1. implementácia vhodne využitelných protokolov referečného modelu TCP/IP,
2. prenosová rýchlosť.

2.1 Analýza sieťového rozhrania

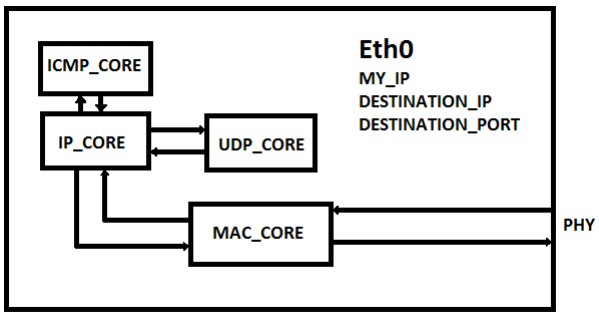
Ako základ generátoru je preto použitý funkčný modul sieťovej komunikácie[19]. Odtestovaných bolo niekoľko projektov, no od začiatku bolo jasné, že vývoj hardvéru sa od vývoja softvéru veľmi líši. Prvým obmedzujúcim faktorom je prepoklad kódu v jazyku VHDL, pretože má mierne lepšie vyjadrovacie schopnosti ako Verilog. Druhým závažným problémom bola neexistencia dostupných modulov univerzálnej sieťovej komunikácie implementovatelná na zariadeniach rodiny Spartan6 bez akýchkoľvek závislostí či už vo forme knižníc alebo prídavných prvkov na FPGA karte, aj napriek tomu že sú veľmi populárne. Toto pramení z komplexity a časovej náročnosti implementácie, s ktorou sa na rozdiel od softvérových vývojárov, kde sú otvorené zdroje populárne, vývojári hardvéru neradi delia. Firmy ako Xilinx alebo Intel ponúkajú implementácie sieťových komunikácii ako produkty, ktorých cena sa pohybuje v tisícoch eur za licenciu. Dostupnejšie riešenia sú znova veľmi komplexné a predstavujú skôr blackbox ako otvorený a upravitelný kód. Amatérska povaha projektov sa odzrkadluje chybovosťou kódov, stručnou no vo väčšine prípadov žiadnou dokumentáciou, neaktívnymi vývojármi, projektami v rôznom stave implementácie, alebo naopak absurnou komplexitou niekoľko tisíc riadkových kódov, v ktorej sa začiatočník nevyzná.

Sieťové rozhranie implementuje protokoly fyzickej, spojovej, sieťovej a transportnej

vrstvy. Bloková schéma na obrázku 2.1 zobrazuje protokolový zásobník implementovaný na zariadení zatiaľ čo obrázok 2.2 zachytáva reláciu medzi obsiahnutými modulmi.



Obr. 2.1: Protokolový zásobník.



Obr. 2.2: Bloková schéma modulov rozhrania.

Zariadenie je schopné príjmať aj odosielať dáta rýchlosťou maximálne 1000Mbit za sekundu.

2.1.1 Príjmanie dát

Dáta sú ako prvé prijaté z fyzického rozhrania do modulu MAC (Media Access Control), ktorý dokáže rozoznať rýchlosť komunikácie medzi 10Mbps, 100MBps alebo 1000Mbps v tvare IP paketu. Ďalej sú dáta 8 bitovou zbernicou posielané do ďalšieho modulu ktorý cyklickou redundantnou kontrolou skontroluje integritu dát. V tomto module sa hlavne kontroluje či je paket určený pre FPGA, v prípade že je mac adresa zhodná s adresou karty alebo broadcastu.

V ďalšej fáze sú dáta poslané do modulu, ktorý predstavuje sieťovú vrstvu referenčného TCP/IP modelu. Tu sa celý paket rozparsuje a vyťažia sa z neho informácie o zdrojovej adrese, cieľovej adrese a použitom protokole vyššej vrstvy a posunie dáta najvyššiemu modulu, ktorý rozhoduje o tom, či sú dáta určené transportnej vrstve alebo modulu ICMP. Dáta sú následne posunuté do poslednej transportnej vrstvy alebo ICMP modulu. V transportnej vrstve sa z UDP paketu extrahujú dáta a prepočítava kontrolný súčet. ICMP modul slúži na odoslanie ICMP Echo Reply.

2.1.2 Odosielanie dát

V opačnom smere sú dáta odoslané následovným spôsobom: V transportnej vrstve sú vstupom samotné dáta, cieľový port a IP adresa cieľa a vzpočíta sa kontrolný súčet. Týmto sa zostaví UDP hlavička a všetky dáta putujú do sieťovej vrstvy. V sieťovej vrstve sa pomocou protokolu ARP overuje či je zaridiadenie identifikované cielovou adresou v spoločnej sieti. Ďalej sú naše dáta spolu s MAC a IP hlavičkami odoslané do linkovej vrstvy. Modul sieťovej vrstvy vypočíta ďalší kontrolný súčet a prípadne dáta vyplní nulami aby spĺňali požadovanú dĺžku.

Spojová vrstva

Modul reprezentujúci fyzickú vrstvu je nazvaný *mac_core*. Pozostáva z dvoch komponentov, prímača *mac_rx* a vysielača *mac_tx*.

Prímacia časť v procese riadenom nábežnou hranou hodinového signálu prímacej domény inicializuje príjmací buffer a základný stav stavového automatu zodpovedného za načítanie dát. Proces implementujúci statový automat má za úlohu:

- orezanie preambule v hlaviče prijatej z fyzického rozhrania,
- kontrolu mac adresy v prijatom pakete, či odpovedá mac adrese čipu alebo broadcastovej adrese,
- kontrolu integrity
- zahadzovanie špatných rámcov
- odosielanie overených rámcov do vyššej vrstvy.

Na obrázoku 2.3 je znázornené overovanie mac adresy.

Vysielacia časť rovnako ako príjmacia je riadená vlastným hodinovým signálom čím si vytvára vlastnú hodinovú doménu. Ako prvý sa spúšťa proces, ktorý načítava dáta z vyššej vrstvy do bufferu. Ďalej stavový automat zabezpečuje funkcie odosielania rámcov aj jumbo rámcov. Rámce s dĺžkou menšou ako 60 bytov doplní nulami na požadovanú minimálnu 60 bytovú veľkosť. Zabezpečuje pridanie CRC32 kontrolného výpočtu. Dáta sú vysielané iba v prípade, že nenastáva kolízia s inými dátami.

```
when st_destin =>
   rx_next_state <= st_destin;
   rx_count_en <= '1';
   calc_crc <= '1';
   case rx_count(3 downto 0) is
      when "0101" =>
      when "0110" =>
         -- Overeni MAC adresy (vcetne broadcastu)
         if (data_i(4) & data_i(3) & data_i(2) & data_i(1) & data_i(0) & gmii_rxd_i = x"FFFFFFFFFFFF") OR
            (data_i(4) & data_i(3) & data_i(2) & data_i(1) & data_i(0) & gmii_rxd_i = FPGA_MAC) then
            rx_next_state <= st_data;
            rx_data_vld_i <= '1';
         else rx_next_state <= st_error; end if;
      when others =>
   end case;
   if GMII_RXER = '1' then rx_next_state <= st_error;end if;
```

Obr. 2.3: Overenie príjmateľa na základe MAC adresy.

Sieťová vrstva

Sieťovú vrstvu implementuje modul *ip_core*. Rovnako ako predchádzajúci modul, pozostáva z príjmacej a vysielacej časti označených *ip_rx* pre prímač z nižšej vrstvy a *ip_tx* pre vysielač dát do nižšej vrstvy.

Prímač je v doméne hodín prímača RX_CLK. Dáta prijaté z mac_rx klasifikuje podla typu na ARP dotaz, ARP odpoved alebo IP paket. Po prijatí celého nevadného rámca alebo paketu indikuje platnosť dát a ich odoslanie do ďalšej vrstvy signálom rx_data_good a rx_data_vld.

Na druhej strane vysielač dátového toku spĺňa iné úlohy. Na začiatku sú tri procesy:

1. LOAD_ADDR1 detekuje v dátach obdržaných z ip_rx informáciu o type obdržaného paketu. Ak sa jedná o ARP dotaz alebo odpoveď, prevezme ich ip adresu a mac adresu do signálov ARP_IP_CACHE a ARP_MAC_CACHE.
2. LOAD_ADDR2 nadstavuje príznak pre odoslanie ARP dotazu.
3. LOAD_ADDR3 nadstavuje príznak pre odoslanie ARP odpoveďe.

Túto funkcionalitu zabezpečujú procesy uvedené na obrázku 2.4.

Hlavnou úlohou tohto komponentu je odosielanie ARP odpovedí a požiadavkov na základe komunikácie s ip_rx a pridávanie IP hlavičky dátam prichádzajúcim z vyššej vrstvy.

Modul *ICMP_core* taktiež patrí do sieťovej vrstvy. Na rozdiel od ostatných komponentov sa neskladá z príjmača a vysielača ale jednoducho spracováva dáta z IP modulu a rovnako upravené dáta aj posiela tomuto modulu naspäť. Proces akým je ICMP paket spracovaný spočíva v načitaní dát z modulu ip_core a v momente

```
LOAD_ADDR1: process (RX_CLK)
begin
   if (RX_CLK'event and RX_CLK = '1') then
      if ARP_REPLY = '1' or ARP_REQUEST = '1' then
         mac_cache_i <= ARP_MAC_CACHE;
         ip_cache_i <= ARP_IP_CACHE;
      end if;
   end if;
end process;


LOAD_ADDR2: process (RX_CLK, RESET, tx_state)
begin
   if RESET = '1' or tx_state = st_reply then
      arp_request_i <= '0';
   elsif RX_CLK'event and RX_CLK = '1' then
      if ARP_REQUEST = '1' then arp_request_i <= '1'; end if;
   end if;
end process;

LOAD_ADDR3: process (RX_CLK, RESET, tx_state)
begin
   if RESET = '1' or tx_state = st_reply then
      arp_reply_i <= '0';
   elsif RX_CLK'event and RX_CLK = '1' then
      if ARP_REPLY = '1' then arp_reply_i <= '1'; end if;
   end if;
end process;
```

Obr. 2.4: Procesy zodpovedné za ARP komunikáciu.

obdržania bytu s hodnotou `x"08"` vieme, že sa jedná a ICMP echo požiadavok. Všetky prijaté dáta sú uložené do pamäte a budú použité pre vytvorenie odpovede. Statový automat si načítava dáta z RAM a odosiela ich modulu ip_tx. Ako prvý byte je odoslaná hodnota `x"00"` reprezentujúca typ ICMP služby, v tomto prípade odpovede. Následne sú odoslané dáta z RAM vrátane prepočítaného kontrolného súčtu.

Transportná vrstva

Implementovaným zástupcom protokolu transportnej vrstvy z dvojice TCP a UDP je protokol UDP. Modul *udp_core* zahŕňa opäť príjmač označený *udp_rx* a vysielač *udp_tx*.

Príjmač UDP je prepojený so sieťovou vrstvou a preberá z nej dáta. Z dát vyhodnocuje cielový port a preberá dáta na výpočet kontrolného súčtu. Datagramy ktoré niesu určené nášmu zariadeniu táto vrstva odhadzuje.

Vysielač transportnej vrstvy obsahuje buffer, do ktorého sa ukladajú dáta určené k odoslaniu. Spolu s informáciami ako cieľová ip adresa, cieľový port, ip adresa zdroja sa vypočíta kontrolný súčet ktorý je súčasťou UDP hlavičky. Pred odosla-

ním prebieha výmena hodnout ako zdrojový port so sieťovou vrstvou. Po odoslaní hlavičky UDP sa predajú dáta z bufferu nižšej vrstve.

Aplikačné rozhranie

Modul *Eth0* zastáva rolu internetového rozhrania do, ktorého je možné zasielať samotné dáta a spolu s parametrami ako cieľová IP adresa dáta putujú celým zásobníkom až na fyzický nosič. Tento modul zapúzdruje a združuje všetky ostatné moduly do jedného funkčného celku. Jeho hlavnou úlohou je rozhodnúť zda prijaté dáta patria do icmp_core alebo do udp_core ako ukazuje obrázok 2.5. Modul Eth0 pristupuje k hodnotám dát v sieťovej vrstve.

```
RX_DMUX: process (source_protocol, d_from_ip, d_from_ip_vld)
begin
   d_to_icmp <= x"00";
   d_to_icmp_vld <= '0';
   d_to_udp <= x"00";
   d_to_udp_vld <= '0';

   case source_protocol is
      when x"01" =>     -- icmp
         d_to_icmp <= d_from_ip;
         d_to_icmp_vld <= d_from_ip_vld;

      when x"11" =>     -- UDP
         d_to_udp <= d_from_ip;
         d_to_udp_vld <= d_from_ip_vld;
      when others =>    -- necinny stav
   end case;
end process;
```

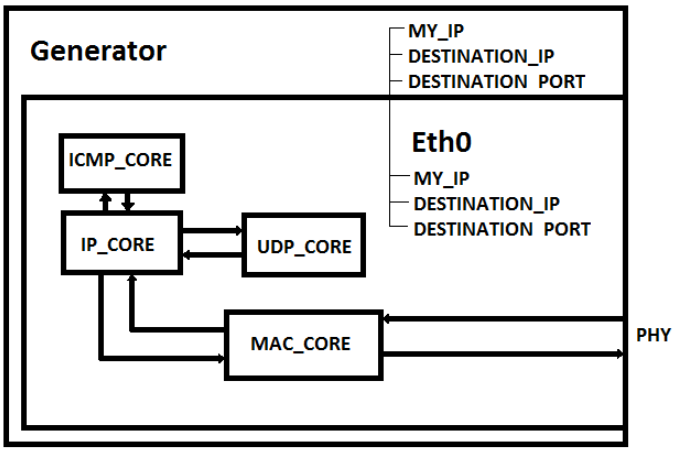
Obr. 2.5: Logika triedenia dát do nižších vrstiev.

2.2 Generátor

Generátor záťaže je modul s názvom *generator* obsahujúci všetky ostatné moduly v tvare komponentov vrátane Eth0. Generovanie dát nastáva každým hodinovým tikom s frekvenciou 125 MHz. Dáta odoslané z modulu generator putujú celým protokolovým zásobníkom, pričom každá vrstva pridá svoje hlavičky. Rozdiel medzi legitýmnym tokom dát a útokom je v obsahu hlavičiek paketov. Nadstavbu modulu *generator* si je možné predstaviť ako obrázok 2.6.

2.2.1 UDP Flood

V prípade UDP záplavy, každý paket má inú zdrojovú IP adresu a cieľový port. Tieto požiadavky sú zabezpečené výpočtom IP adresy na začiatku procesu. Hodnoty IP adries sa pohybujú v rozmedzí 192.168.1.2 - 223.255.255.254 cyklicky. Následujúci



Obr. 2.6: Bloková schéma upraveného rozhrania.

výpis kódu (2.7) ukazuje dynamickú povahu zdrojovej IP adresy. Na začiatku sa kontroluje spúštací príznak zahájenia generovania START_DOS, ktorý je vo východzom stave zapnutý.

```
gen: process(gtxclk,START_DOS,gen_TX_CLK,gen_RX_CLK,gen_PHY_TXCLK,gen_PHY_GTXCLK) is
begin
 if rising_edge(gtxclk) then
 if RESET = '1' then
          START_DOS <= '0';
      end if;
 if START_DOS = '1' then
--hodnota je priradena nizsie v procese MY_IP <= last_src_ip
 if src_ip_oc1 < x"FE" then
   src_ip_oc1 <= std_logic_vector(unsigned(src_ip_oc1) + 1);
   last_src_ip <= src_ip_oc4 & src_ip_oc3 & src_ip_oc2 & src_ip_oc1;
elsif src_ip_oc1 = x"FE" then
      if src_ip_oc2 < x"FF" then
      src_ip_oc2 <= std_logic_vector(unsigned(src_ip_oc2) + 1);
      src_ip_oc1 <= x"01";
      last_src_ip <= src_ip_oc4 & src_ip_oc3 & src_ip_oc2 & src_ip_oc1;

      elsif src_ip_oc2 = x"FF" then
         if src_ip_oc3 < x"FF" then
         src_ip_oc3 <= std_logic_vector(unsigned(src_ip_oc3) + 1);
         src_ip_oc2 <= x"01";
         src_ip_oc1 <= x"01";
         last_src_ip <= src_ip_oc4 & src_ip_oc3 & src_ip_oc2 & src_ip_oc1;

         elsif src_ip_oc3 = x"FF" then
            if src_ip_oc4 < x"DF" then
            src_ip_oc4 <= std_logic_vector(unsigned(src_ip_oc4) + 1);
            src_ip_oc3 <= x"01";
            src_ip_oc2 <= x"01";
            src_ip_oc1 <= x"01";
            last_src_ip <= src_ip_oc4 & src_ip_oc3 & src_ip_oc2 & src_ip_oc1;
            elsif src_ip_oc4 = x"C0" then
               last_src_ip <= default_src_ip;
```

Obr. 2.7: Dynamicky premenlivá zdrojová IP adresa.

Kus kódu zaručujúci premenlivé čísla portov v rozsahu od 1 do 65000 je zachytený na obrázku 2.8. Bude existovať niekoľko portov nezasiahnutých útokom ale to je maličkosť. Hlavné je, že dáta nesmerujú na jediný port ale na väčšinu portov. V prípade, že číslo portu dosiahne hodnotu 6500, zresetuje sa znova na 1.

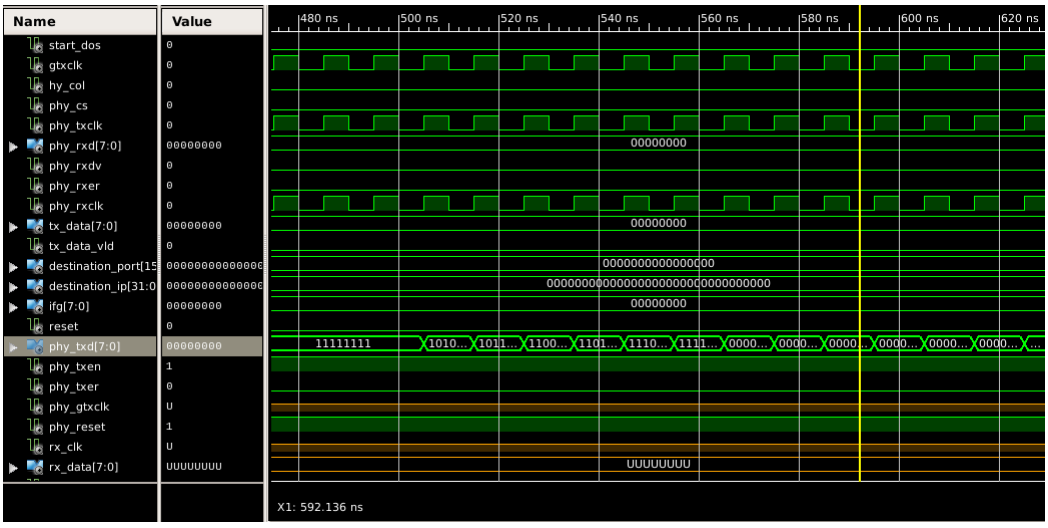
```
gen_MY_IP <= last_src_ip;
nextvalue := x"0000";
nextvalue := std_logic_vector(unsigned(port_number) + 1 );
port_number := nextvalue;
if port_number < x"FDE8" then

gen_DESTINATION_IP <= x"0A1402DB";
gen_DESTINATION_PORT <= port_number;
gen_TX_DATA <= x"AA";
gen_TX_DATA_VLD <= '1';


else
port_number := x"0000";
end if;
```

Obr. 2.8: Dynamicky premenlivé číslo portu.

Treba rátať s tým, že cieľová adresa je daná staticky v zdrojovom kóde.

Simulácia

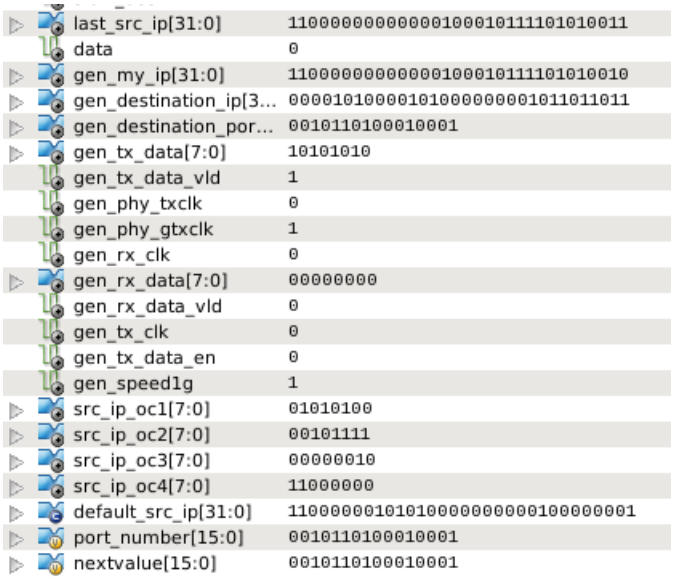
Nižšie uvedené obrázky zachytávajú výsledné správanie generátora UDP záplavy. Na obrázku 2.9 je najdôležitejším parametrom phy_txd, teda fyzický výstup dát. Jasne vidíme, že dáta tečú linkou.



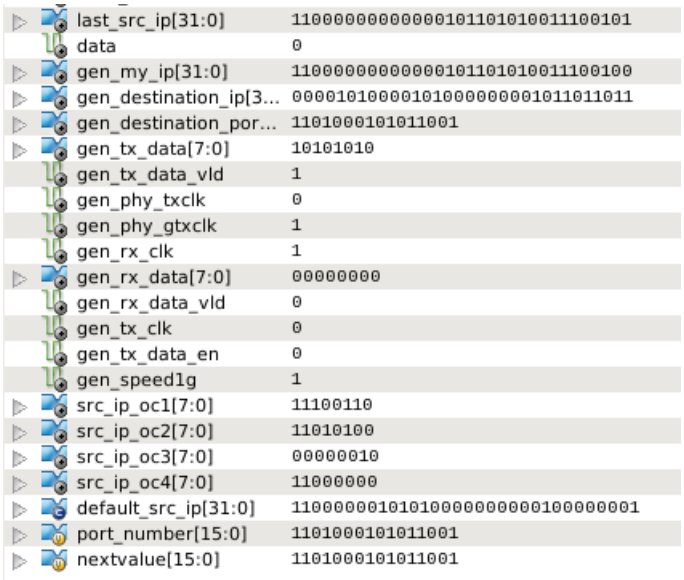
Obr. 2.9: Simulácia UDP paketu.

Na nasledujúcich obrázkoch (2.10 a 2.11) sú zachytené v náhodných časoch premenné, čo dokazuje ich modalitu:

- gen_MY_IP,
- gen_destination_port,
- gen_destination_ip,
- iné ich pomocné premenné.



Obr. 2.10: Premenné v čase t1.



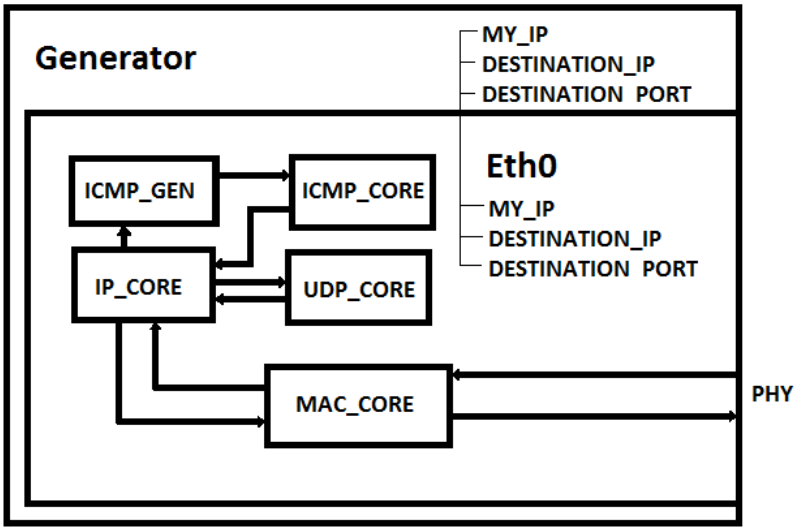
Obr. 2.11: Premenné v čase t2.

2.2.2 ICMP Ping záplava

Ďalším útokom je ICMP Ping záplava, ktorá funguje na rovnakej báze ako UDP záplava. Rozdiel je v použitom protokole a odozvy a štýlu spracovania ICMP paketov cieľom. V útoku je použitý typ správy ICMP Echo Request, známy ako ping, čo núti cieľ útoku odpovedať. Generátor ICMP pakety typu odpoveď zahadzuje, cieľ si však zahltí odpoveďami celú širku pásma.

Návrh

Bloková schéma návrhu (obrázok 2.12):



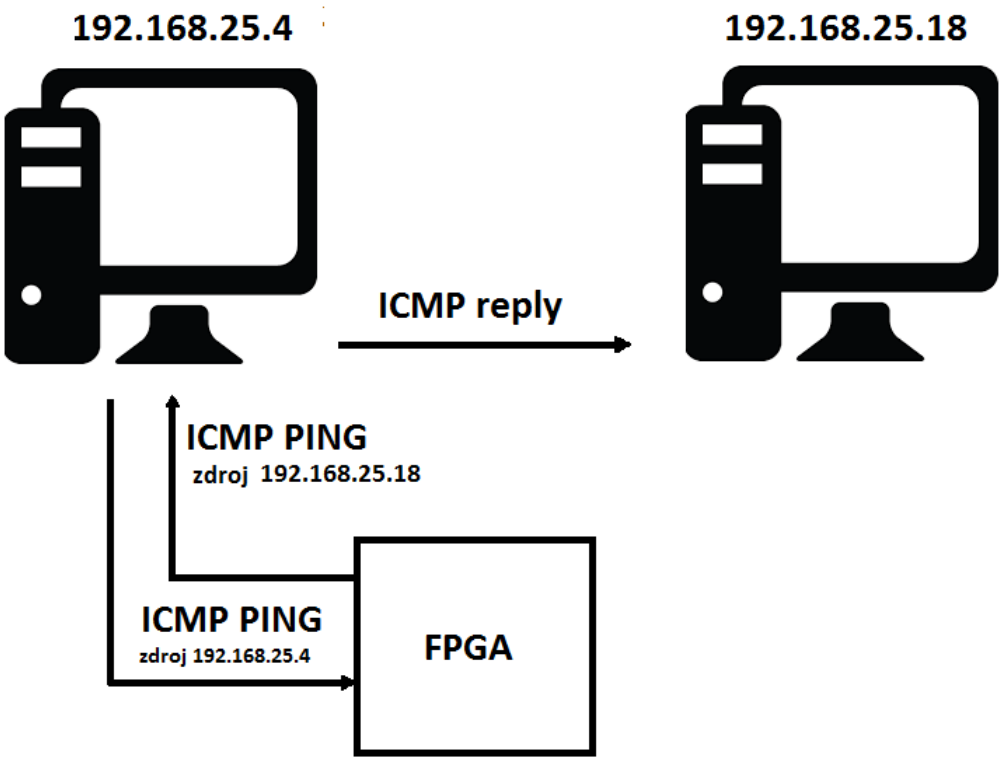
Obr. 2.12: Bloková schéma generátoru ICMP Ping záplavy.

Spúštačom útoku je počiatočný ICMP ping obete. Tento paket zachytíme v sieťovej vrstve v prechode z modulu ip_rx do modulu icmp_core. Modul icmp_core je upravený tak aby neodosielal naspať ICMP Echo Reply, ale vlastný ICMP Echo Ping pôvodnému odosielateľovy. Zmena dotazu na dotaz je na obrázku 2.13

Modul *icmp_gen* pôvodný spúštací paket zachztí a spustí proces v sebe proces, ktorý bude tento paket neustále odosielať modulu icmp_core na spracovanie, čo vyvolá záplavovú situáciu. V prípade, že by sme poznali IP adresu ďalšieho zariadenia, zdrojová IP adresa ICMP paketov z FPGA by mohla byť upravená. Tak by sme vyvolali reťazovú reakciu, kedy by cieľ útoku posielal odpovede ďalšiemu zariadeniu. Popísanu situáciu znázorňuje obrázok 2.14.

```
icmp_tx: process (tx_state, checksum_calc, ram_output,
                  write_address, read_address, reply, DATA_TO_IP_ACK)
begin
   read_en <= '0';
   reply_rst <= '0';
   data_to_ip_vld_i <= '0';
   data_to_ip_i <= x"08";  --povodne x"00" 8 = echo request

   case tx_state is
      ------------------------------------------------------------------
      -- Start -----------------------------------------------------------
      when st_idle =>
         tx_next_state <= st_idle;
         if reply = '1' then
            tx_next_state <= st_wait_ack;
         end if;
```

Obr. 2.13: Transformácia ICMP paketu.



Obr. 2.14: Návrh zlepšenia generátoru ICMP záplavy.

3 Záver

Cieľom práce bolo zoznámenie sa s platformou FPGA a návrh generátoru záťaže kybernetickými DoS útokmi za použitia jazyka VHDL. V prvej časti práce je popísaný význam FPGA čipov, ich vlastnosti a využitie. Popisuje aj rozdelenie a architektúru FPGA, čo dáva dobý prehľad pri vyberaní vhodej karty na implementáciu generátoru záťaže.

K návrhu bol použitý jazyk VHDL, ktorý je rovnako ako proces vývoja hardvérového dizajnu popísaný v prvej kapitole. Práca sa detailnejšie venuje popisu referenčných modelov ISO/OSI a TCP/IP, ktoré sú základom prenosu dát medzi zariadeniami všetkých telekomunikačných a počítačových sietí. Bližšie sú popísané protokoly pracujúce s dátami na jednotlivých vrstvách modelov, menovite protokoly: IP, UDP a ICMP, ktoré sú neskôr využité pri implementácii generátoru záťaže. V poslednej časti prvej kapitoly je spomenuté množstvo DoS útokov z ktorých boli dva implementované.

Kriticou časťou generátora záťaže na ktorej je založený je implementované sieťové rozhranie s funkčnou sieťovou komunikáciou, ktoré v sebe zahŕňa protokoly ARP, IP, ICMP a UDP. Časťou práce bolo aj hladanie vhodného projektu na rozšírenie, no to bolo veľmi časovo náročné a len z časti úspešné. Projekt použitý ako základ implementuje sieťové rozhranie s maximálnym dátovým tokom 1 Gigabit, a je otázne či by táto šírka pásma stačila na zahltenie napadnutého zariadenia vzhľadom na to, že sieťové karty v osobných počítačoch dokážu spracovávať dátový tok o veľkosti 1 Gigabit.

Modul generátor je nadstavbou a rozšírením funkcionality tohto projektu. Projekt bol upravený tak, aby sa premávka ním generovaná dala považovať za DoS útok. Útok typu UDP Flood je odsimulovaný vo vývojovom prostredí Xilinx ISE. Práca obsahuje návrh modulu realizujúci útok typu ICMP Flood a v projekte nastali modifikácie potrebné aby bol tento útok realizovatelný.

Literatúra

[1] WOLF, Wayne. *FPGA based system design.* New Jersey: Prentice-Hall, 2004. ISBN 01-314-2461-0.

[2] *Project Catapult* [online]. [cit. 2018-12-11]. Dostupné z: <https://www.microsoft.com/en-us/research/project/project-catapult/>

[3] *Global FPGA Market* [online]. In: . [cit. 2019-05-27]. Dostupné z: <https://www.grandviewresearch.com/industry-analysis/fpga-market>

[4] *Preffered FPGA Dev's Vendor's* [online]. In: . [cit. 2019-05-27]. Dostupné z: <https://blogs.synopsys.com/breakingthethreelaws/2013/12/xilinx-fpga%E2%80%99s-for-fpga-based-prototyping/>

[5] *FPGA Architectures Overview.* In: Portland State University [online]. [cit. 2018-12-11]. Dostupné z: <https://www.pdx.edu/nanogroup/sites/www.pdx.edu.nanogroup/files/FPGA-architecture.pdf>

[6] HAMDY. In: GREENE, Johnatan, Esmat HAMDY a Sam BEAL. *Antifuse Field Programmable Gate Arrays* [online]. [cit. 2018-12-11]. Dostupné z: <https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=231343&tag=1>

[7] *Xilinx Spartan-6 Family Overview* [online]. In: . [cit. 2019-05-27]. Dostupné z: <https://www.xilinx.com/support/documentation/data_sheets/ds160.pdf>

[8] PINKER, Jiří a Martin POUPA. *Číslicové systémy a jazyk VHDL.* Praha: BEN - technická literatura, 2006. ISBN isbn80-7300-198-5.

[9] *RFC 790. ASSIGNED NUMBERS* [online]. In: . [cit. 2019-05-27]. Dostupné z: <https://tools.ietf.org/html/rfc790>

[10] *RFC 768. User Datagram Protocol* [online]. In: . [cit. 2019-05-27]. Dostupné z: <https://tools.ietf.org/html/rfc768>

[11] *RFC 791. INTERNET PROTOCOL* [online]. In: . [cit. 2019-05-27]. Dostupné z: <https://tools.ietf.org/html/rfc791>

[12] *RFC 792. INTERNET CONTROL MESSAGE PROTOCOL* [online]. In: . [cit. 2019-05-27]. Dostupné z: <https://tools.ietf.org/html/rfc792>

[13] *ISO Reference Model for Open Systems Interconnection (OSI)* [online]. In: . [cit. 2019-05-27]. Dostupné z: <http://www.bitsavers.org/pdf/datapro/communications_standards/2783_ISO_OSI.pdf>

[14] *KUROSE, James F. Computer networking: a top-down approach featuring the Internet. 6.* Boston: Addison-Wesley, c2001. ISBN 0-201-47711-4.

[15] *The Largest DDoS attacks of all time* [online]. In: . [cit. 2018-12-14]. Dostupné z: <https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/>

[16] KHALIMONENKO, Alexander, Oleg KUPREEV a Ekaterina BADOVSKAYA. *DDoS attacks in Q1 2018* [online]. 26.6. [cit. 2018-12-11]. Dostupné z: <https://securelist.com/ddos-report-in-q1-2018/85373/

[17] DDOS ATTACKS. *DDoS Knowledge Center - Latest Threats and Mitigation Methods | Incapsula* [online]. [cit. 2018-12-11]. Dostupné z: <https://www.incapsula.com/ddos/ddos-attacks.html>

[18] *DDOS-GUARD Attack types* [online]. [cit. 2019-05-27]. Dostupné z: <https://ddos-guard.net/en/terminology/attack_type>

[19] P. Skibik *Zdrojový kód sieťového rozhrania* [online]. [cit. 2019-05-27]. Dostupné z: <https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=40494>

Zoznam symbolov, veličín a skratiek

FPGA	Field programable gate array
ASIC	Application-specific integrated circuit
HDL	Hardware description language
VHDL	Very high speed integrated circuit hardware description language
TCP	Transmission control protocol
UDP	User datagram protocol
HTTP	Hyper text transmission protocol
ICMP	Internet controll message protocol
IPv4	Internet protocol version 4
DoS	Denial of service
DDoS	Distributed denial of service

Zoznam príloh

A Príloha 1

V prílohe je s prácou odovzdaný zdrojový kód sieťového rozhrania spolu s generátorom UDP záplavy, ktorý bol vytvorený v rámci praktickej časti tejto práce a ich testovacie súbory.